TECHNICAL RESEARCH REPORT

Evolutionary Policy Iteration for Solving Markov Decision Processes

by Hyeong Soo Chang, Hong-Gi Lee, Michael Fu, and Steven Marcus

TR 2002-31

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2002		**00-00-2002 to 00-00-2002**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Evolutionary Policy Iteration for Solving Markov Decision Processes	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Maryland,Electrical Engineering Department,Institute for Systems Research,College Park,MD,20742**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**12**	

Evolutionary Policy Iteration for Solving Markov Decision Processes

Hyeong Soo Chang, Hong-Gi Lee, Michael C. Fu*, and Steven I. Marcus
Institute for Systems Research
University of Maryland, College Park, MD 20742
E-mail: {hyeong,mfu,marcus}@isr.umd.edu, hglee@cau.ac.kr

July 16, 2002

Abstract

We propose a novel algorithm called Evolutionary Policy Iteration (EPI) for solving infinite horizon discounted reward Markov Decision Process (MDP) problems. EPI inherits the spirit of the well-known PI algorithm but eliminates the need to maximize over the entire action space in the policy improvement step, so it should be most effective for problems with very large action spaces. EPI iteratively generates a "population" or a set of policies such that the performance of the "elite policy" for a population is monotonically improved with respect to a defined fitness function. EPI converges with probability one to a population whose elite policy is an optimal policy for a given MDP. EPI is naturally parallelizable and along this discussion, a distributed variant of PI is also studied.

Keywords: (Distributed) policy iteration, Markov decision process, genetic algorithm, evolutionary algorithm, parallelization

*Corresponding author.
Michael C. Fu can be reached by telephone at +1 (301) 405-2241 or by fax at +1 (707) 897-3774.
This work was supported in part by the National Science Foundation under Grant DMI-9988867, in part by the Air Force Office of Scientific Research under Grant F496200110161, and in part by the Department of Defense under Contract MDA 90499C2521.

1 Introduction

We propose a novel algorithm called Evolutionary Policy Iteration (EPI) to solve Markov Decision Processes (MDPs) for an infinite horizon discounted reward criterion. The algorithm is especially targeted to problems where the state space is relatively small but the action space is extremely large, so that the policy improvement step in Policy Iteration (PI) becomes computationally impractical. EPI eliminates the operation of maximization over the entire action space in the policy improvement step by directly manipulating policies via a method called "policy switching" [1] that generates an improved policy from a set of given policies. The computation time for generating such an improved policy is on the order of the state space size. The basic algorithmic procedure imitates that of standard genetic algorithm (GAs) (see, e.g., [5] [10] [9]) with appropriate modifications and extensions required for the MDP setting, based on an idea similar to the "elitism" concepts introduced by De Jong [3]. In our setting, the elite policy for a population is a policy obtained via policy switching that improves the performances of all policies in the population. EPI starts with a set of policies or "population" and converges with probability one (w.p. 1) to a population of which the elite policy is an optimal policy, while maintaining a certain monotonicity property for elite policies over generations with respect to a fitness value.

The literature applying evolutionary algorithms such as GAs for solving MDPs is relatively sparse. The recent work of Lin, Bean, and White [6] uses a GA approach to construct the minimal set of affine functions that describes the value function in partially observable MDPs, yielding a variant of value iteration. Chin and Jafari [2] propose an approach that maps heuristically "simple" GA [10] into the framework of PI. However, their evolutionary operations do not include policy switching, and convergence to an optimal policy is not always guaranteed.

As noted earlier, the main motivation for the proposed EPI algorithm is the setting where the action state space is finite but extremely large. In this case, it could be computationally impractical to apply exact PI or value iteration, due to the requirements of maximization over the entire action space via e.g., enumeration or random search methods. On the other hand, local search cannot guarantee that a global maximum has been found. Thus, the monotonicity in the policy improvement step is not preserved. The proposed EPI algorithm preserves an analogous monotonicity property over the elite policies in the populations.

A primary contribution of our work is the use of a (random) evolutionary search algorithm in the context of MDPs with a convergence guarantee (w.p. 1). Another contribution is the development of a parallelizable algorithm for solving MDP problems exactly via policy switching. We partition the policy space with nonoverlapping subsets of the policy space and then apply EPI or PI into each subset in parallel. Distributed EPI applies policy switching to (convergent) elite policies for the subsets, obtaining an optimal policy for the original policy space (see Section 4).

This note is organized as follows. We start with the problem setting and necessary background

on MDPs in Section 2. In Section 3, we formally describe the EPI algorithm with detailed discussion and present the convergence proof. In Section 4, we study a distributed variant of PI and discuss how to speed up EPI by parallelization. We then conclude with some remarks in Section 5.

2 Background

Consider an MDP with finite state space X, finite action space A, reward function $R : X \times A \to \mathcal{R}$, and transition function P that maps a state and action pair to a probability distribution over X. We denote the probability of transitioning to state $y \in X$ when taking action a in state $x \in X$ by $P(x,a)(y)$. For simplicity, we assume that every action is admissible in every state.

Let Π be the set of all stationary policies $\pi : X \to A$. Define the *optimal value* associated with an initial state $x \in X$:

$$\begin{aligned} V^*(x) &= \max_{\pi \in \Pi} V^\pi(x), x \in X, \text{ where} \\ V^\pi(x) &= E\left[\sum_{t=0}^{\infty} \gamma^t R(x_t, \pi(x_t)) \middle| x_0 = x\right], x \in X, 0 < \gamma < 1, \pi \in \Pi, \end{aligned}$$

where x_t is a random variable denoting state at time t and γ is the discount factor. Throughout the paper, we assume that γ is fixed. The problem we address is that of finding an optimal policy π^* that maximizes the *expected* optimal value for an initial state distributed with probability distribution δ, i.e.,

$$\pi^* \in \arg\max_{\pi \in \Pi} E\left[V^\pi(X_0)\right], \quad X_0 \sim \delta. \tag{1}$$

Policy iteration (PI) can be used to solve (1). For a given initial state, PI computes an optimal policy in a finite number of steps, because there are a finite number of policies in Π, and PI preserves the monotonicity in terms of the policy performance. The PI algorithm consists of two parts: policy evaluation and policy improvement. Let $B(X)$ be the space of real-valued bounded measurable functions on X. We define an operator $T : B(X) \to B(X)$ as

$$T(\Phi)(x) = \max_{a \in A} \left\{ R(x,a) + \gamma \sum_{y \in X} P(x,a)(y)\Phi(y) \right\}, \Phi \in B(X), x \in X, \tag{2}$$

and similarly, an operator $T_\pi : B(X) \to B(X)$ for $\pi \in \Pi$ as

$$T_\pi(\Phi)(x) = R(x, \pi(x)) + \gamma \sum_{y \in X} P(x, \pi(x))(y)\Phi(y), \Phi \in B(X), x \in X. \tag{3}$$

It is well known (see, e.g., [8]) that for each policy $\pi \in \Pi$, there exists a corresponding unique $\Phi \in B(X)$ such that for $x \in X$,

$$T_\pi(\Phi)(x) = \Phi(x) \text{ and } \Phi(x) = V^\pi(x).$$

The policy evaluation step obtains V^π for a given π via (3), and the policy improvement step obtains $\hat{\pi} \in \Pi$, e.g., as the argument of the right-hand side of (2), such that

$$T(V^\pi)(x) = T_{\hat{\pi}}(V^\pi)(x), x \in X.$$

The policy $\hat{\pi}$ improves π in that $V^{\hat{\pi}}(x) \geq V^\pi(x)\ \forall x \in X$. However, carrying out the policy improvement step may be impractical for large A, motivating our EPI algorithm as an alternative.

3 Evolutionary Policy Iteration

3.1 Algorithm description

As with all evolutionary/GA algorithms, we define the kth generation population ($k = 0, 1, 2, ...$), denoted by $P(k)$, which is a set of policies in Π, and $n = |P(k)| \geq 2$ is the population size, which we take to be constant in each generation. Given the fixed initial state probability distribution δ defined over X, we define the *average value of π for δ* or *fitness value of π*:

$$J_\delta^\pi = \sum_{x \in X} V^\pi(x)\delta(x).$$

Note that J_δ^π is simply the expectation given by the function on the right-hand side of (1), and an *optimal policy* π^* satisfies

$$J_\delta^{\pi^*} \geq J_\delta^\pi \quad \forall \pi \in \Pi.$$

A high-level description of the EPI algorithm is shown in Figure 1, where some steps (e.g., mutation) are described at a conceptual level, with details provided in the following subsections. We denote P_m as the mutation selection probability, P_g the global mutation probability, and P_l the local mutation probability. We also define an *action selection distribution* μ as a probability distribution over A such that $\sum_{a \in A} \mu(a) = 1$ and $\mu(a) > 0$ for all $a \in A$.

3.2 Initialization and Policy Selection

Convergence of the EPI algorithm is independent of the initial population $P(0)$ (to be shown later), mainly due to the **Policy Mutation** step. We can randomly generate an initial population or start with a set of heuristic policies. One simple initialization is a population of policies with the property that the same action is prescribed for every state, but each policy in the population prescribes a different action.

3.3 Policy Switching

One of the basic procedural steps in GA is to select members from the current population to create a "mating pool" to which "crossover" is applied; this step is called "parent selection". Similarly,

Evolutionary Policy Iteration (EPI)

- **Initialization:**
  Select population size n and $K > 0$. $P(0) = \{\pi_1, ..., \pi_n\}$, where $\pi_i \in \Pi$.
  Set $N = k = 0$, and P_m, P_g and P_l in $(0, 1]$, and $\pi^*(-1) = \pi_1$.
- **Repeat:**
  - **Policy Switching:**
    * Obtain V^π for each $\pi \in P(k)$.
    * Generate the elite policy of $P(k)$ defined as
    $$\pi^*(k)(x) \in \{\arg\max_{\pi \in P(k)}(V^\pi(x))(x)\}, x \in X.$$
    * **Stopping Rule:**
      · If $J_\delta^{\pi^*(k)} \neq J_\delta^{\pi^*(k-1)}$, $N = 0$.
      · If $J_\delta^{\pi^*(k)} = J_\delta^{\pi^*(k-1)}$ and $N = K$, terminate EPI.
      · If $J_\delta^{\pi^*(k)} = J_\delta^{\pi^*(k-1)}$ and $N < K$, $N \leftarrow N + 1$.
    * Generate $n-1$ random subsets $S_i, i = 1, ..., n-1$ of $P(k)$ by selecting $m \in \{2, ..., n-1\}$ with equal probability and selecting m policies in $P(k)$ with equal probability.
    * Generate $n-1$ policies $\pi(S_i)$ defined as:
    $$\pi(S_i)(x) \in \{\arg\max_{\pi \in S_i}(V^\pi(x))(x)\}, x \in X.$$
  - **Policy Mutation:** For each policy $\pi(S_i), i = 1, ..., n-1$,
    * Generate a "globally" mutated policy $\pi^m(S_i)$ w.p. P_m using P_g and μ or a "locally" mutated policy $\pi^m(S_i)$ w.p. $1 - P_m$ using P_l and μ.
  - **Population Generation:**
    * $P(k+1) = \{\pi^*(k), \pi^m(S_i)\}$, $i = 1, ..., n-1$.
    * $k \leftarrow k + 1$.

Figure 1: Evolutionary Policy Iteration (EPI)

we can design a "policy selection" step to create a mating pool; there are many ways of doing this. The **Policy Switching** step includes this selection step implicitly.

Given a nonempty subset Δ of Π, we define a policy $\bar{\pi}$ generated by *policy switching* with respect to Δ as

$$\bar{\pi}(x) \in \{\arg\max_{\pi \in \Delta}(V^\pi(x))(x)\}, x \in X. \quad (4)$$

For completeness, we show that the policy generated by policy switching improves any policy in Δ (see also Theorem 3 in [1].)

Theorem 3.1 *Consider a nonempty subset* Δ *of* Π *and the policy* $\bar{\pi}$ *generated by policy switching*

with respect to Δ *given in Equation (4). Then, for all* $x \in X$,

$$V^{\bar{\pi}}(x) \geq \max_{\pi \in \Delta} V^{\pi}(x).$$

Proof: We begin with a lemma, which states a basic property of the T_π-operator defined by (3).

Lemma 3.1 *Given* $\pi \in \Pi$, *suppose there exists* $\Phi \in B(X)$ *for which*

$$T_\pi(\Phi)(x) \geq \Phi(x), x \in X. \tag{5}$$

Then, $V^\pi(x) \geq \Phi(x)$ *for all* $x \in X$.

Proof: By successive applications of the T_π-operator to both sides of Equation (5) and the monotonicity property of the operator, we have that for all $x \in X$,

$$\lim_{n \to \infty} T_\pi^n(\Phi)(x) \geq \Phi(x).$$

And by the Banach fixed point theorem, $\lim_{n\to\infty} T_\pi^n(\Phi)(x) = V^\pi(x), x \in X$, which proves the lemma. ∎

Now define $\Phi(x) = \max_{\pi \in \Delta} V^\pi(x)$ for all $x \in X$. Pick an arbitrary state $x \in X$. From the definition, there exists a policy $\pi' \in \Delta$ such that $V^{\pi'}(x) \geq V^\pi(x)$ for all $\pi \in \Delta$ and $\bar{\pi}(x) = \pi'(x)$. It follows that

$$\begin{aligned} T_{\bar{\pi}}(\Phi)(x) &= R(x, \bar{\pi}(x)) + \gamma \sum_{y \in X} P(x, \bar{\pi}(x))(y)\Phi(y) \\ &= R(x, \pi'(x)) + \gamma \sum_{y \in X} P(x, \pi'(x))(y)\Phi(y) \\ &\geq R(x, \pi'(x)) + \gamma \sum_{y \in X} P(x, \pi'(x))(y)V^{\pi'}(y) \\ &= V^{\pi'}(x) = \Phi(x). \end{aligned}$$

By the lemma above, the claim is proved. ∎

The above theorem immediately implies the following result.

Corollary 3.1 *Consider a nonempty subset* Δ *of* Π *and the policy* $\bar{\pi}$ *generated by policy switching with respect to* Δ *given in Equation (4). Then, for any initial state distribution* δ,

$$J_\delta^{\bar{\pi}} \geq \max_{\pi \in \Delta} J_\delta^\pi.$$

Proof: From the definition of J_δ^π for $\pi \in \Pi$,

$$J_\delta^{\bar{\pi}} = \sum_{x \in X} V^{\bar{\pi}}(x)\delta(x) \geq \sum_{x \in X} \max_{\pi \in \Delta} V^\pi(x)\delta(x) \geq \max_{\pi \in \Delta} \sum_{x \in X} V^\pi(x)\delta(x) = \max_{\pi \in \Delta} J_\delta^\pi,$$

where the last inequality follows from Jensen's inequality. ∎

We first generate a policy $\pi^*(k)$, called the elite policy with respect to the current population $P(k)$, which improves any policy in $P(k)$ via policy switching. Note that this is different from the elitist concept of De Jong [3], where the elitist is the best policy in $P(k)$. EPI includes the elite policy generated by policy switching unmutated in the the new population. By doing so, the population contains a policy that improves any policy in the previous population. Therefore, the following monotonicity property holds:

Lemma 3.2 *For any δ and for all $k \geq 0$,*

$$J_\delta^{\pi^*(k)} \geq J_\delta^{\pi^*(k-1)}.$$

Proof: The proof is by induction. The base step is obvious from the definition of $\pi^*(0)$ and $\pi^*(-1)$ by Corollary 3.1. Assume that $J_\delta^{\pi^*(i)} \geq J_\delta^{\pi^*(i-1)}$ for all $i \leq k$. Because the EPI algorithm includes $\pi^*(k)$ in $P(k+1)$, the elite policy at $k+1$ is generated over a population that contains $\pi^*(k)$, which implies that $J_\delta^{\pi^*(k+1)} \geq J_\delta^{\pi^*(k)}$. ∎

We then generate $n-1$ random subsets $S_i, i = 1, ..., n-1$ of $P(k)$ as follows. We first select $m \in \{2, ..., n-1\}$ with equal probability and then select m policies from $P(k)$ with equal probability. By applying policy switching, we generate $n-1$ policies defined as

$$\pi(S_i)(x) \in \{\arg\max_{\pi \in S_i}(V^\pi(x))(x)\}, x \in X.$$

These policies will be mutated to generate a new population (see the next subsection).

The policy switching step is a key part in EPI to speed up the convergence of EPI. Suppose that S_i for some i consists of two policies π_1 and π_2 and let $\Psi = \{x | \pi_1(x) \neq \pi_2(x), x \in X\}$. Write $\pi > \pi'$ if for all $x \in X$,

$$V^\pi(x) \geq V^{\pi'}(x)$$

and for some state $x \in X$,

$$V^\pi(x) > V^{\pi'}(x)$$

and write $\pi \geq \pi'$ if for all $x \in X$, $V^\pi(x) \geq V^{\pi'}(x)$. Then there are at least $|\Psi|$ policies $\hat{\pi}_j$, $j = 1, ..., |\Psi|$ such that for each j, either

$$\pi(S_i) \geq \hat{\pi}_j > \pi_1 \text{ or } \pi(S_i) \geq \hat{\pi}_j > \pi_2$$

holds. In other words, by one application of policy switching, we eliminate at least $|\Psi|$ policies but at most $|X|$ in the search process. This is because given a policy π, if we can improve the policy π by modifying the actions in m states, we rule out at least m policies that are better than π. See Lemma 5 [7] for a formal proof.

As we can see, policy switching directly manipulates policies to generate an improved policy relative to all policies it was applied to, eliminating the operation of maximization over the entire action space, which is the main computational advantage that replaces the policy improvement step in the original PI.

3.4 Policy Mutation

Policy mutation is carried out by altering a given policy in the following manner: for each state, the currently prescribed action is replaced probabilistically. The main reason for mutating policies is to avoid being caught in a local maximum, making a probabilistic convergence guarantee possible (see the convergence proof below). We consider two types of mutation: "local" and "global", which are distinguished by the degree of mutation, as indicated by the number of states with changed actions in the mutated policy. To this end, we assume that $P_l \ll P_g$, with P_l being close to zero and P_g being close to one. The **Policy Mutation** step first determines whether a given policy π is mutated globally or locally, using Bernoulli probability P_m. If π is globally (resp., locally) mutated, then for each state x, $\pi(x)$ is changed w.p. P_g (resp., P_l), where the action to which it is changed would follow the given action selection distribution μ. Local mutation helps the algorithm fine-tune good policies via local search, whereas global mutation helps the algorithm escape from local maximum. One simply way to select the particular action to which the current action is mutated is to select randomly (uniformly) among all other actions.

3.5 Population Generation and Stopping Rule

At each kth generation, the new population $P(k+1)$ is simply given by the elite policy generated from $P(k)$ and $n-1$ mutated policies from $\pi(S_i), i = 1, ..., n-1$. This population generation method allows a policy that is poor in terms of performance, but might be in the neighborhood of an optimal value located at the top of the very narrow hill, to be kept in the population so that a new search region can be started from the policy. This helps the algorithm to avoid being caught in the region of local optima.

Once we have a new population, we need to test whether EPI should terminate. Even if the fitness values for the two consecutive elite policies are identical, this does not necessarily mean that the elite policy is an optimal policy as in PI. Therefore, we run the EPI algorithm K more times so that these random jumps by the mutation step will eventually bring EPI to a neighborhood of the optimum. As the value of K gets larger, the probability of being in a neighborhood of the optimum increases. Therefore, the elite policy at the termination is the right policy with more confidence as K increases.

3.6 Convergence

Theorem 3.2 *Given $P_m > 0$, $P_g > 0$, $P_l > 0$, and an action selection distribution μ such that $\sum_{a \in A} \mu(a) = 1$ and $\mu(a) > 0 \ \forall a \in A$, $\pi^*(k) \longrightarrow \pi^*$ w.p. 1 as $K \to \infty$ for any $P(0)$.*

Proof: The proof is straightforward. Observe first that as $K \to \infty$, $k \to \infty$. This is because EPI terminates when $N = K$ and if $N \neq K$, the value of k increases by one.

From the assumption, the probability of generating an optimal policy by the **Policy Mutation** step is positive. To see this, let α be the probability of generating one of the optimal policies by local mutation and let β the probability of generating one of the optimal policies by global mutation. Then,

$$\begin{aligned} \alpha &\geq \prod_{x\in X} P_l\mu(\pi^*(x)) = (P_l)^{|X|}\cdot\prod_{x\in X}\mu(\pi^*(x)) > 0 \\ \beta &\geq \prod_{x\in X} P_g\mu(\pi^*(x)) = (P_g)^{|X|}\cdot\prod_{x\in X}\mu(\pi^*(x)) > 0, \end{aligned} \tag{6}$$

where π^* is a particular optimal policy in Π. Therefore, the probability of generating an optimal policy by the **Policy Mutation** step is positive and this probability is independent of $P(0)$.

Therefore, the probability that $P(k)$ does not contain an optimal policy (starting from an arbitrary $P(0)$) is at most $((1-\alpha)P_m))^{(n-1)k}((1-\beta)(1-P_m))^{(n-1)k}$, which goes to zero as $k\to\infty$. By Lemma 3.2, once $P(k)$ contains an optimal policy, $P(k+m)$ contains an optimal policy for any $m\geq 1$ because the fitness value of an optimal policy is the maximum among all policies in Π. This proves the claim. ∎

4 Parallelization

The EPI algorithm can be naturally parallelized and by doing so, we can improve the running rate. Basically, we partition the policy space Π into subsets of $\{\Pi_i\}$ such that $\bigcup_i \Pi_i = \Pi$ and $\Pi_i\cap\Pi_j = \emptyset$ for all $i\neq j$. We then apply EPI to each Π_i in parallel, and then once each part terminates, the best policy π_i^* from each part is taken. We then apply policy switching to the set of best policies $\{\pi_i^*\}$. We state a general result regarding parallelization of any algorithm that finds optimal polices for MDPs.

Theorem 4.1 *Given a partition of* Π *such that* $\bigcup_i \Pi_i = \Pi$ *and* $\Pi_i\cap\Pi_j=\emptyset$ *for all* $i\neq j$*, consider an algorithm* $\mathcal{A}$ *that generates the best policy* π_i^* *for* Π_i *such that for all* $x\in X$*,*

$$V^{\pi_i^*}(x) \geq \max_{\pi\in\Pi_i} V^{\pi}(x).$$

Then, the policy $\bar{\pi}$ *defined as*

$$\bar{\pi}(x) \in \{\arg\max_{\pi_i^*}(V^{\pi_i^*}(x))(x)\}, x\in X,$$

is an optimal policy for Π*.*

Proof: Via policy switching, $\bar{\pi}$ improves the performance of each π_i^*, i.e.,

$$V^{\bar{\pi}}(x) \geq \max_{\pi_i^*} V^{\pi_i^*}(x), x\in X,$$

implying that $\bar{\pi}$ is an optimal policy for Π, since the partition covers the entire policy space. ∎

Note that we cannot just pick the best policy among π_i^* in terms of the fitness value J_δ^π. The condition that $J_\delta^\pi \geq J_\delta^{\pi'}$ for $\pi \neq \pi'$ does not always imply that $V^\pi(x) \geq V^{\pi'}(x)$ for all $x \in X$ even though the converse is true. In other words, we need a policy that improves all policies π_i^*. Picking the best policy among such policies does not necessarily guarantee an optimal policy for Π.

If the number of subsets in the partition is N, the overall convergence of the algorithm $\mathcal{A}$ is faster by a factor of N. For example, if at state x, the action a or b can be taken, let $\Pi_1 = \{\pi | \pi(x) = a, \pi \in \Pi\}$ and $\Pi_2 = \{\pi | \pi(x) = b, \pi \in \Pi\}$. By using this partition, the convergence rate of the algorithm $\mathcal{A}$ will be twice as fast.

By Theorem 4.1, this idea can be applied to PI via policy switching, yielding a "distributed" PI. We apply PI to each Π_i. Once PI for each part terminates, we combine the resulting policy for each part by policy switching. The combined policy is an optimal policy so that this method will speed up the original PI by a factor of N if the number of subsets in the partition is N. However, note that this distributed variant of PI will also involve the operation of the maximization over the action space in the policy improvement step. The result of Theorem 4.1 also naturally extends to a *dynamic programming version* of PI, similarly to EPI. For example, we can partition Π by Π_1 and Π_2, and Π_1 is subdivided by Π_{11} and Π_{12}, and Π_2 by Π_{21} and Π_{22}. The optimal substructure property is preserved by policy switching. Suppose that the number of subsets generated in this way is β, then the overall computation time of an optimal policy is $O(\beta \cdot |X| \cdot C)$, where C is the maximum size of the subsets in terms of the number of policies, because policy switching is applied $O(\beta)$ times with $O(|X|)$ complexity and C is the upper bound on PI-complexity.

5 Concluding Remarks

The discussion in the previous section raises an important question that can motivate further research: How can we partition the policy space so that PI or EPI converges faster? For well-chosen partitions, we may even be able to obtain optimal policies for some subsets *analytically*. Much of the MDP literature concentrates on aggregation in the state space (see, e.g., [4]) for an approximate solution for a given MDP. Our discussion on the parallelization of PI and EPI can be viewed in some sense as an aggregation in the policy space, where the distributed version of EPI can be used to generate an approximate solution of a given MDP.

In our setting, the mutated action for a mutated state was determined (probabilistically) by a given action selection distribution. If the action space is continuous, say $[0, 1]$, a straightforward implementation might change only the least significant digit for local mutation and the most significant digit for global mutation, where numbers in $[0, 1]$ are represented by a certain number of significant digits.

GAs are known to work well for many continuous domain problems but to face difficulties of a different kind for problems where the decision variables are discrete [9]. However, EPI circum-

vents this problem via policy switching, an idea that has not been exploited in the GA literature previously.

References

[1] H. S. Chang, R. Givan, and E. K. P. Chong, "Parallel rollout for on-line solution of partially observable Markov decision processes," *Discrete Event Dynamic Systems: Theory and Application*, submitted, 2000.

[2] H. Chin and A. Jafari, "Genetic algorithm methods for solving the best stationary policy of finite Markov decision processes," in *Proc. of the 30th Southeastern Symposium on System Theory*, 1998, pp. 538–543.

[3] K. A. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems,* Ph.D. Thesis, Univ. of Michigan, Ann Arbor, MI, 1975.

[4] R. Givan, S. Leach, and T. Dean, "Bounded Markov decision processes," *Artificial Intelligence*, Vol. 122, pp. 71–109, 2000.

[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning,*, Addison-Wesley, 1989.

[6] A. Z.-Z. Lin, J. Bean, and C. White, III, "A hybrid genetic/optimization algorithm for finite horizon partially observed Markov decision processes," Technical Report 98-25, Dept. of Ind. and Oper. Eng., Univ. of Michigan, Ann Arbor, 1998.

[7] Y. Mansour and S. Singh, "On the complexity of policy iteration," in *Proc. UAI*, 1999, pp. 401–408.

[8] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley, New York, 1994.

[9] C. R. Reeves, "Genetic algorithms for the operations researcher," *INFORMS J. on Computing*, Vol. 9, No. 3, pp. 231–250, 1997.

[10] M. Srinivas, and L. M. Patnaik, "Genetic algorithms: a survey," *IEEE Computer*, Vol. 27. No. 6, pp. 17–26, 1994.